Internet-Draft
<draft-satran-iscsi-01.txt>
Expires January 10, 2001

J. Satran
D. Smith
K. Meth
IBM

C. Sapuntzakis
Cisco Systems

Randy Haagens
Hewlett-Packard Co.

Efri Zeidner
SANGate

Paul Von Stamwitz
Adaptec

Luciano Dalle Ore
Quantum

July 10, 2000

iSCSI (Internet SCSI)

Status of this Memo

Acknowledgements

A large group of people contributed through their review, comments and valuable insights to the creation of this document - too many to mention them all. Nevertheless, we are grateful to all of them. We are especially grateful to those that found the time and

patience to participate in our weekly phone conferences and intermediate meetings in Almaden and Haifa and thus helped shape this document: Matt Wakeley (Agilent), Jim Hafner, John Hufferd, Prasenjit Sarkar, Meir Toledano, John Dowdy, Steve Legg, Alain Azagury (IBM), Dave Nagle (CMU), David Black (EMC), John Matze (Veritas), Mark Bakke, Steve DeGroote, Mark Shrandt (NuSpeed), Gabi Hecht

Exhibit C

(Gadzoox), Robert Snively (Brocade), Nelson Nachum (StorAge)

Table of Contents

1. Abstract

The Small Computer Systems Interface (SCSI) is a popular family of protocols for communicating with I/O devices, especially storage devices. This memo describes a transport protocol for SCSI that operates on top of TCP. The iSCSI protocol aims to be fully compliant with the requirements laid out in the SCSI Architecture Model - 2 [SAM2] document.

2. Overview

2.1. SCSI Concepts

The endpoint of most SCSI commands is a "logical unit" (LU). Examples of logical units include hard drives, tape drives, CD and DVD drives, printers and processors. Within the logical unit the abstract entity that executes the SCSI commands is named the device-server. A "target" is a collection of logical units, in general of the same kind, and is directly addressable on the network. In large installations a target is known also as a "control unit". The target corresponds to the server in the abstract SAM client-server model. An "initiator" creates and sends SCSI commands to the target. The initiator corresponds to the client in the abstract SAM client-server model. A "task" is a linked set of SCSI commands. Some LUNs support multiple pending (queued) tasks. The target uses a "task tag" to distinguish between tasks. Only one command in a task can be outstanding at any given time. A SCSI command results in an optional data phase and a response phase. In the data phase, information travels either from the initiator to the target, as in a WRITE command, or from target to initiator, as

in a READ command. In the response phase, the target returns the final status of the operation, including any errors. A response terminates a SCSI command.

2.2. iSCSI Concepts & Functional Overview

2.2.1. Layers & Sessions

The following conceptual layering model is used in this document to specify initiator and target actions and how those relate to transmitted and received Protocol Data Units: - SCSI layer builds/receives SCSI CDB (Command Data Blocks) and relays/receives them with the remaining command execute parameters (cf. SAM-2) to/from the - iSCSI layer that builds/receives iSCSI PDUs and relays/receives them to/from - one or more TCP connections that form an initiator-target "session"

Communication between initiator and target occurs over one or more

TCP connections. The TCP connections are used for sending control messages, SCSI commands, parameters and data within iSCSI Protocol Data Units (iSCSI PDU). The group of TCP connections linking an initiator with a target form a session (loosely equivalent to a SCSI nexus); a session is defined by a session ID (composed of an initiator part and a target part). TCP connections can be added and removed from a session.

2.2.2. Ordering and iSCSI numbering

iSCSI supports ordered command delivery within a session. All commands (initiator-to-target) and responses (target-to-initiator) are numbered. Any SCSI activity is related to a task (SAM-2). The task is identified by the Initiator Task Tag for the life of the task.

Commands in transit from the initiator SCSI layer to the target SCSI layer are numbered by iSCSI and the number is carried by the iSCSI PDU as CmdRN (Command-Reference-Number). All iSCSI PDUs that have a task association carry this number. CmdRNs are allocated by the initiator iSCSI as an increasing counter wrapping around from 2**32-1 to 1. The 0 value is reserved and used to mean immediate delivery. The target may choose to deliver some task management commands for immediate delivery. The means by which the SCSI layer may request immediate delivery for a command or by which iSCSI will decide by itself to mark a PDU for immediate delivery are yet to be defined. CmdRNs are significant only during command delivery to the target. Once the device serving part of the target SCSI has received a command, CmdRN ceases to be significant. The initiator and target are assumed to have three registers that define the allocation mechanism - CmdRN - the current command reference number advanced by 1 on each command shipped; ExpCmdRN - the next expected command by the target - acknowledges all commands up to it; MaxCmdRN - the maximum number to be shipped - defines the queuing capacity of the receiving iSCSI. CmdRN can take any value from ExpCmdRN to MaxCmdRN except 0. The target will reject any command outside this range or duplicates within the range not flagged with the retry bit (the X bit in the opcode). The target and initiator registers are supposed to uphold causal ordering.

Responses in transit from the target to the initiator are also numbered. The StatRN (Status Reference Number) is used for this purpose. If the target uses data packet numbering and all the inbound

data have been acknowledged, or the target is able to regenerate inbound data, then the target may free all the resources allocated for the task execution just after sending a response. The same holds for targets not allowing full command recovery. The result summary, just enough to rebuild the status PDU, will be kept by those iSCSI target implementations that support status recovery

after connection failure. As the only cause for long delays in responses can be failed connections and received responses free-up resources, we felt that score boarding responses at the initiator could be accomplished by simple bitmaps and there is no need to flow-control responses. Status acknowledgment is done by the initiator through ExpStatRN (Expected Status RN) and large difference between StatRN and ExpStatRN indicates a failed connection.

iSCSI initiators are required to implement the numbering scheme if they support more than one connection.

iSCSI targets are not required to use the numbering scheme for ordered delivery even when they support multiple connections. However they are required to provide ExpCmdRN and MaxCmdRN values that will enable the initiator to make progress.

The NOP PDUs are not associated with a task, are meant for immediate delivery, and their only purpose is synchronizing the ordering registers of the target and initiator.

2.2.3. iSCSI Login

The purpose of iSCSI login is to enable a TCP connection for iSCSI use, authenticate the parties, authorize the initiator to send SCSI commands and mark the connection as belonging to a iSCSI session. A session is used to identify to a target all the connections with a given initiator. The targets listen on a well-known TCP port for incoming connections. The initiator begins the login process by connecting to that well-known TCP port. As part of the login process, the initiator and target MAY wish to authenticate each other. This can occur in many different ways. For example, the endpoints may wish to check the IP address of the other party. If the TCP connection uses transport layer security [TLS], certificates may be used to identify the endpoints. Also, iSCSI includes commands for identifying the initiator and passing an authenticator to the target (see Appendix B). Once suitable authentication has occurred, the target MAY authorize the initiator to send SCSI commands. How the target chooses to authorize an initiator is beyond the scope of this document. The target indicates a successful authentication and authorization by sending a login response with "accept login". The login message includes a session ID - composed with an initiator part ISID and a target part TSID. For a new session the TSID is null. As part of the response the target will generate a TSID. Session specific parameters can be specified only for the first login of a session (TSID null)(e.g., the maximum number of connections that can be used for this session). Connection specific parameters (if any) can be specified for any login. Thus a session

is operational once it has at least one connection. After authentication and authorization, other parameters may be negotiated using the highly extensible Text Command message that allows arbitrary key:value pairs to be passed. Any message sent on a TCP connection before this connection gets into full feature phase at the initiator should be rejected by the initiator. A message reaching a target on a TCP connection before the full feature phase will be rejected with an iSCSI check condition.

2.2.4. iSCSI Full Feature Phase

Once the initiator is authorized to do so, the iSCSI session is in iSCSI full feature phase. The initiator may send SCSI commands and data to the various LUNs on the target by wrapping them in iSCSI messages that go over the established iSCSI session. For SCSI commands that require data and/or parameter transfer, the (optional) data and the status for a command must be sent over the same TCP connection that was used to deliver the SCSI command (connection allegiance). Thus if an initiator issues a READ command, the target must send the requested data followed by the status to the initiator over the same TCP connection that was used to deliver the SCSI command. If an initiator issues a WRITE command, the initiator must send the data for that command and the target must return the status over the same TCP connection that was used to deliver the SCSI command. During iSCSI Full Feature Phase, the initiator and target may interleave unrelated SCSI commands, their SCSI Data and responses, over the session. Outgoing SCSI data (initiator to target - user data or command parameters) is sent as either unsolicited data or solicited data. Unsolicited data can be part of an iSCSI command PDU ("immediate data") or an iSCSI data PDU. Solicited data are sent in response to Ready To Transfer PDUs. Targets operate in either solicited (RTT) data mode or unsolicited (non RTT) data mode. An initiator must always honor an RTT data request. It is considered an error for an initiator to send unsolicited data PDUs to a target operating in RTT mode (only solicited data). An initiator is allowed to send immediate data even to targets working in RTT mode. An initiator may request, at login, to send immediate data of any size and a target may indicate the size of immediate data blocks it is ready to accept in its response. A target is allowed to silently discard data and request retransmission through RTT. Initiators will not perform any score boarding for data and the residual count calculation is to be performed by the targets. Incoming data is always solicited. However an initiator will be able to request retransmission of all or part of the target data. SCSI Data packets are matched to their corresponding SCSI commands by using Tags that are specified in the protocol. Initiator tags for pending commands are unique initiator-wide for a session. Target tags for pending commands are unique target-wide

for the session. Although the above mechanisms are designed to accomplish efficient data delivery and a large degree of control over the data flow it is recognized that some specific sequences involving ordered execution and a mix of solicited and immediate data can result in deadlocks. It is for this reason that discarding data by a target is considered a legitimate action. Each iSCSI session to a target is treated as if it originated from a different initiator.

2.2.5. iSCSI Connection Termination

- Connection termination is assumed to be an exceptional event. Graceful TCP connection shutdowns are done by sending TCP FINs. Graceful connection shutdowns MUST only occur when there are no outstanding tasks that have allegiance to the connection. A target SHOULD respond rapidly to a FIN from the initiator by closing its half of the connection as soon as it has finished all outstanding tasks that have allegiance to the connection. Closing a connection that has outstanding tasks may require recovery actions and will Be described elsewhere in this document.

2.2.6. Naming & mapping

Targets are named using an URL type name of the format:

```
scsi://<domain-name>[/modifier]
```

The name used to connect will be optionally included in the login in order to enable the target to present different views. This is the Target Acquired Name (TAN). We will not attempt to define which components of the name will participate in the name resolution process and which ones will be used only for "view" definition. The syntactic sugar included might be used to introduce structure for management purposes but has no specific significance for this standard. Example:

```
scsi://diskfarm1.acme.com
scsi://computingcenter.acme.com/peripherals/diskfarm1
```

When a target has to act as an initiator for a third party command it will use the TAN during login as required by the authentication mechanism. A domain name that contains exactly four numbers separated by dots (.), where each number is in the range 0 through 255, will be interpreted as an IPv4 address. Examples:


```
10.0.0.1/tapefarm1
10.0.0.2
```

Likewise a domain name that contains more than four, but less than 16 numbers separated by dots (.), where each number is in the range 0 through 255, will be interpreted as an Ipv6 address. Examples:

```
12.5.7.10.0.0.1/tapefarm1
12.5.6.10.0.0.2
```

To address targets and logical units within a target SCSI uses a fixed length (8 bytes) uniform addressing scheme; in this document we call those addresses SCSI reference addresses (SRA).

To provide the target with the protocol specific addresses (iSCSI or FC) iSCSI uses a Map Command; the Map command sends the managing target the protocol specific addresses and gets from the target the SRAs to use in subsequent commands. For iSCSI a protocol specific address is a TCP address and a view; those can be expressed as

numeric IPV4 or IPV6 address+port sequences (6 or 18 bytes) followed by a view or in text form. After mapping iSCSI will be provided with a handle to the address in standard SCSI format.

3. Message Formats

All multi-byte integers specified in formats defined in this document are to be represented in network byte order (i.e., big endian).

3.1. Template Header and Opcodes

All iSCSI messages and responses have a header of the same length (48 bytes). Additional data may be added, as necessary, beginning with byte 48. The fields of Opcode and Length appear in all message and response headers. The other most commonly used fields are Initiator Task Tag, Logical Unit Number, and Flags, which, when used, always appear in the same location of the header.

```
Byte /    0          |      1        |       2       |       3       |
    /                |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode        | Opcode-specific fields                        |
   +---------------+---------------+---------------+---------------+
  4| Length of Data (after 48 byte Header)                         |
   +---------------+---------------+---------------+---------------+
  8| LUN or Opcode-specific fields                                 |
   +                                                               +
 12|                                                               |
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20/ Opcode-specific fields                                        /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48
```

3.1.1. Opcode

The Opcode indicates which iSCSI type of message or response is encapsulated by the header. The opcode is further encoded as follows:

```
    b7   Retry Command/Response (X bit)
    b6   Response
    b5-0 Operation
```

Valid opcodes for messages (sent by initiator to target) are:

```
  0x00 NOP-Out Message (from initiator to target)
  0x01 SCSI Command (encapsulates a SCSI Command Descriptor Block)
  0x02 SCSI Task Management Command
```



```
  0x03 Login Command
  0x04 Text Command
  0x05 SCSI Data (for WRITE operation)
  0x09 Ping Command (from initiator to target)
  0x0a Map Command
```

Valid opcodes for responses (sent by target to initiator) are:

```
  0x40 NOP-In Message (from target to initiator)
  0x41 SCSI Response (contains SCSI status and possibly sense
       information or other response information)
  0x42 SCSI Task Management Response
  0x43 Login Response
  0x44 Text Response
  0x45 SCSI Data (for READ operation)
  0x46 Ready To Transfer (RTT - sent by target to initiator when
       it is ready to receive data from initiator)
  0x47 Asynchronous Event (sent by target to initiator to indicate
  certain special conditions)
  0x48 Opcode Not Understood
  0x49 Ping Response (from target to initiator)
  0x4a Map Response
```

3.1.2. Length

The Length field indicates the number of bytes, beyond the first 48 bytes, that are being sent together with this message header. It is anticipated that most iSCSI messages and responses (not counting data transfer messages) will not need more than the 48 byte header, and hence the Length field will contain the value 0. It is expected that larger than 16 byte CDBs and parameter data will follow the header.

3.1.3. LUN

The LUN specifies the Logical Unit for which the command is targeted. If the command does not relate to a Logical Unit, this field is either ignored or may be used for some other purpose. According to [SAM2], a Logical Unit Number can take up to a 64-bit field that identifies the Logical Unit within a target device. The exact format of this field can be found in the [SAM2] document.

3.1.4. Initiator Task Tag

The initiator assigns a Task Id (or tag) to each SCSI task that it issues. (Recall that a task is a linked set of SCSI commands.)

This Tag is an initiator-wide unique identifier that can be used to uniquely identify the Task.

3.1.5. Opcode-specific fields

These field have different meanings for different messages.

3.2. SCSI Command

```
Byte /    0           |       1       |       2       |       3       |
    /                 |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x01) |I I|R| Rsrvd   |A| Rsvrd |ATTR | Reserved (0)  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8| Logical Unit Number (LUN)                                     |
   +                                                               +
 12|                                                               |
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Expected Data Transfer Length                                 |
   +---------------+---------------+---------------+---------------+
 24| CmdRN                                                         |
   +---------------+---------------+---------------+---------------+
 28| ExpStatRN                                                     |
   +---------------+---------------+---------------+---------------+
 32/ SCSI Command Descriptor Block (CDB)                           /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48/ Additional Data (Command Dependent)                           /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

3.2.1. Flags.

The flags field for a SCSI Command consists of two bytes.

```
     Byte 1 - iSCSI flags
     b7-6 (I) Immediate Data from initiator to target (command
          parameters/write/control); this field indicates also how
          to interpret the length field:
        00 - Immediate Data Length = Length; CDB Length = 16
        01 - CDB Length = Length+16; Immediate Data Length = 0
        10 - Immediate Data Length = Length 24 MSB; CDB Length = 16
             + Length(8 LSB)
        11 - Immediate Data Length = Length 16 MSB; CDB Length = 16
             + Length(16 LSB)
     b5   (R) set when data is expected to flow from target to ini-
          tiator (read).
     b0-4 Reserved (should be 0)
     Byte 2 - SCSI flags
```



```
     b7   (A) set to turn off Autosense for this command (see
          [SAM2]).
     b3-6 Reserved (should be 0)
     b0-2 used to indicate Task Attributes.
```

Autosense refers to the automatic return of sense data to the initiator in case a command did not complete successfully. If autosense is turned off, the initiator must explicitly request that sense data be sent to it after some command has completed with a CHECK CONDITION status.

3.2.2. Task Attributes

The Task Attribute field (ATTR) can have one of the following

integer values (see [SAM2] for details):

0 Untagged
1 Simple
2 Ordered
3 Head of Queue
4 ACA

3.2.3. Command Reference Number (CmdRN) enables ordered delivery

3.2.4. ExpStatRN - Expected Status Reference Number

Acknowledges status. Responses up to ExpStatRN -1 (mod 2**32) have been received. This number will also update the internal register. Values that do not appear as "increasing" will be ignored; this may be required when updates arrive out of order (they travel on different TCP connections).

3.2.5. Expected Data Transfer Length

The Expected Data Transfer Length field states the number of bytes of data that the initiator expects will be sent for this (READ or WRITE) SCSI operation in SCSI Data packets. For a WRITE operation, the initiator uses this field to specify the number of bytes of data it expects to transfer for this operation in data packets (not counting data headers). For a READ operation, the initiator uses this field to specify the number of bytes of data it expects the target to transfer to the initiator (not counting data headers).

If no data will be transferred in SCSI Data packets for this SCSI operation, this field should be set to 0.

Upon completion of a data transfer, the target will inform the initiator of how many bytes were actually processed (sent or received) by the target.

3.2.6. SCSI Command Descriptor Block (CDB)

There are 16 bytes in the CDB field, designed to accommodate the largest currently defined CDB. If, in the future, larger CDBs are allowed, the spill-over of the CDB may extend beyond the 48-byte header.

3.2.7. Command-Data

Some SCSI commands require additional parameter data to accompany the SCSI command. This data may be placed beyond the 48-byte boundary of the iSCSI header. Alternatively user data can be placed in the same PDU (in both cases we talk about immediate data).

3.3. SCSI Response

```
Byte /    0          |       1         |       2         |       3         |
    /                |                 |                 |                 |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x41) | Rsvd(0)   |O|U| Reserved (0)                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8| Reserved (0)                                                  |
   +                                                               +
 12|                                                               |
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Residual Count                                                |
   +---------------+---------------+---------------+---------------+
 24| StatRN                                                        |
   +---------------+---------------+---------------+---------------+
 28| ExpCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 32| MaxCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 36| Command Status|iSCSI Status   | Reserved (0)                  |
   +---------------+---------------+---------------+---------------+
 40| Res_len                       | Sense_len                     |
   +---------------+---------------+---------------+---------------+
 44| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 48/ Response and/or sense Data (optional)                         /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

3.3.1. Flags

Byte 1

b0 (U) set for Residual Underflow. In this case, the Residual Count indicates how many bytes were not transferred out of those expected to be transferred.

b1 (O) set for Residual Overflow. In this case, the Residual Count indicates how many bytes could not be transferred because the initiator's Expected Data Transfer Length was too small.

b2-7 not used (should be set to 0).

Bits 0 and 1 are mutually exclusive.

3.3.2. Residual Count

The Residual Count field is valid only in case either the Residual Underflow bit or Residual Overflow bit is set. If neither bit is set, the Residual Count field should be 0. If the Residual Underflow bit is set, the Residual Count indicates how many bytes were not transferred out of those expected to be transferred. If the Residual Overflow bit is set, the Residual Count indicates how many bytes could not be transferred because the initiator's Expected Data Transfer Length was too small.

3.3.3. Command Status

The Command Status field is used to report the SCSI status of the command (as specified in [SAM2]).

3.3.4. iSCSI Status

The iSCSI Status field is used to report the status of the command before it was sent by the target to the LUN. The values are given below.

0 Good status
1 iSCSI check

If the iSCSI field is not 0 the command status will indicate CHECK CONDITION

3.3.5. Res_len - Response length

3.3.6. Sense_len - Sense length

3.3.7. Response or Sense Data

If Autosense was not disabled in the originating CDB and the Command Status was CHECK CONDITION (0x02), then the Response Data field will contain sense data for the failed command. Some sense codes will relate to iSCSI check conditions (e.g. excessive number of outstanding commands, immediate data blocks too large etc.). If the Command Status is Good (0x00) then the Response Data field will contain data from the data phase of the CDB. The Length parameter specifies the number of bytes in this field. If no error occurred, and no data is needed for the response to the SCSI Command the Length field is 0. Note that if the Command Status was CHECK CONDITION but Autosense was disabled, then sense data must be explicitly requested by the initiator with a new SCSI command.

3.3.8. StatRN - Status Reference Number

StatRN is a reference number that the target iSCSI layer generates whenever it issues a response by incrementing an internal counter. A gap in StatRN indicates a lost status (possible due to connection failure).

3.3.9. ExpCmdRN - next expected CmdRN from this initiator

This field will be used to update the internal register but values not between the current value of the ExpCmdRN and MaxCmdRN are ignored; this may be required when updates arrive out of order (they travel on different TCP connections)

3.3.10. MaxCmdRN - maximum CmdRN acceptable from this initiator

This field will be used to update the internal register but values not between the current value of the MaxCmdRN and ExpCmdRN are ignored; this may be required when updates arrive out of order (they travel on different TCP connections)

Update order is MaxCmdRN, ExpCmdRN

3.4. NOP-Out Message

```
Byte /    0          |         1         |         2         |         3         |
    /                |                   |                   |                   |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
```

```
    +---------------+---------------+---------------+---------------+
   0| Opcode (0x00) |P| Reserved (0)| Reserved (0)                  |
    +---------------+---------------+---------------+---------------+
   4| Length                                                        |
    +---------------+---------------+---------------+---------------+
   8/ Reserved (0)                                                  /
   +/                                                               /
    +---------------+---------------+---------------+---------------+
  28| ExpStatRN                                                     |
    +---------------+---------------+---------------+---------------+
  32/ Reserved (0)                                                  /
   +/                                                               /
    +---------------+---------------+---------------+---------------+
  48
```

3.4.1. P - poll bit

Request a NOP-In message

3.5. NOP-In Message

```
  Byte /    0          |       1       |       2       |       3       |
      /                |               |               |               |
     |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
    +---------------+---------------+---------------+---------------+
   0| Opcode (0x40) |P| Reserved (0)| Reserved(0)                   |
    +---------------+---------------+---------------+---------------+
   4| Length                                                        |
    +---------------+---------------+---------------+---------------+
   8/ Reserved (0)                                                  /
   +/                                                               /
    +---------------+---------------+---------------+---------------+
  28| ExpCmdRN                                                      |
    +---------------+---------------+---------------+---------------+
  32| MaxCmdRN                                                      |
    +---------------+---------------+---------------+---------------+
```

```
   36/ Reserved (0)                                                  /
    +/                                                               /
     +---------------+---------------+---------------+---------------+
   48
```

3.5.1. P - poll bit

Request a NOP-Out message

3.6. Asynchronous Event

An Asynchronous Event may be sent from the target to the initiator without corresponding to a particular command. The target specifies the status for the event and sense data.

```
   Byte /      0        |       1       |       2       |       3       |
       /                |               |               |               |
      |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
      +---------------+---------------+---------------+---------------+
     0| Opcode (0x47) | Reserved (0)                                  |
      +---------------+---------------+---------------+---------------+
     4| Length                                                        |
      +---------------+---------------+---------------+---------------+
     8| Logical Unit Number (LUN)                                     |
      +                                                               +
    12|                                                               |
      +---------------+---------------+---------------+---------------+
    16/ Reserved (0)                                                  /
     +/                                                               /
      +---------------+---------------+---------------+---------------+
    24| StatRN                                                        |
      +---------------+---------------+---------------+---------------+
    28| ExpCmdRN                                                      |
      +---------------+---------------+---------------+---------------+
    32| MaxCmdRN                                                      |
      +---------------+---------------+---------------+---------------+
    36|SCSI Event Ind |iSCSI Event Ind| Reserved (0)                  |
      +---------------+---------------+---------------+---------------+
    40/ Reserved (0)                                                  /
```

```
   /                                                               /
   +---------------+---------------+---------------+---------------+
 48/ Sense Data                                                    /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

3.6.1. iSCSI Event

Some Asynchronous Events are strictly related to iSCSI while others are related to SAM-2. The codes returned for iSCSI Asynchronous Events are:

2 Target is being reset.

3.6.2. SCSI Event Indicator

The following values are defined. (See [SAM2] for details.)

1 An error condition was encountered after command completion.
2 A newly initialized device is available.
3 Some other type of unit attention condition has occurred.
4 An asynchronous event has occurred.

Sense Data accompanying the report identifies the condition. The Length parameter is set to the length of the Sense Data.

3.7. SCSI Task Management Command

```
Byte /      0         |        1        |        2        |        3        |
    /                 |                 |                 |                 |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x02) | Function      | Reserved (0)                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8| Logical Unit Number (LUN)                                     |
   +                                                               +
 12|                                                               |
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Referenced Task Tag or Reserved (0)                           |
   +---------------+---------------+---------------+---------------+
 24| CmdRN                                                         |
   +---------------+---------------+---------------+---------------+
 28| ExpStatRN                                                     |
   +---------------+---------------+---------------+---------------+
 32/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48
```

3.7.1. Function

The Task Management functions provide an initiator with a way to explicitly control the execution of one or more Tasks. The Task Management functions are summarized as follows (for a more detailed description see the [SAM2] document):

1 Abort Task---aborts the task identified by the Referenced Task Tag field.
2 Abort Task Set---aborts all Tasks issued by this initiator on the Logical Unit.
3 Clear ACA---clears the Auto Contingent Allegiance condition.
4 Clear Task Set---Aborts all Tasks (from all initiators) for the Logical Unit.
5 Logical Unit Reset.
6 Target Reset.

For the functions above except <Target Reset>, a SCSI Task Management Response is returned, using the Initiator Task Tag to identify the operation for which it is responding. For the <Target Reset>

function, the target cancels all pending operations. The target may send an Asynchronous Event to all attached initiators notifying them that the target is being reset. The target then closes all of its TCP connections to all initiators (all sessions are terminated).

3.8. SCSI Task Management Response

```
 Byte /      0        |       1       |       2       |       3       |
     /                |               |               |               |
    |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
    +---------------+---------------+---------------+---------------+
   0| Opcode (0x42) | Response      | Reserved (0)                  |
    +---------------+---------------+---------------+---------------+
   4| Length                                                        |
```

```
     +---------------+---------------+---------------+---------------+
    8| Logical Unit Number (LUN)                                     |
     +                                                               +
   12|                                                               |
     +---------------+---------------+---------------+---------------+
   16| Initiator Task Tag                                            |
     +---------------+---------------+---------------+---------------+
   20| Reserved (0)                                                  |
     +---------------+---------------+---------------+---------------+
   24| StatRN                                                        |
     +---------------+---------------+---------------+---------------+
   28| ExpCmdRN                                                      |
     +---------------+---------------+---------------+---------------+
   32| MaxCmdRN                                                      |
     +---------------+---------------+---------------+---------------+
   36| Response      | Reserved (0)                                  |
     +---------------+---------------+---------------+---------------+
   40/ Reserved (0)                                                  /
    +/                                                               /
     +---------------+---------------+---------------+---------------+
   48
```

For the functions <Abort Task, Abort Task Set, Clear ACA, Clear Task Set, Logical Unit reset>, the target performs the requested Task Management function and sends a SCSI Task Management Response back to the initiator. The target includes all of the information the initiator provided in the SCSI Task Management Message, so the initiator can know exactly which SCSI Task Management Message was serviced. In addition, the target provides a Response which may take on the following values:

     0   Function Complete
     1   Function Rejected

For the <Target Reset> function, the target cancels all pending operations. The target may send an Asynchronous Event to all attached initiators notifying them that the target is being reset. The target then closes all of its TCP connections to all initiators (terminates all sessions).

3.9. Ready To Transfer (RTT)

When an initiator has submitted a SCSI Command with data passing from the initiator to the target (WRITE), the target may specify which blocks of data it is ready to receive. In general, the target may request that the data blocks be delivered in whatever order is convenient for the target at that particular instant. This information is passed from the target to the initiator in the Ready To Transfer (RTT) message. In order to allow write operations without RTT, the initiator and target must have agreed to do so by both sending the UseRTT:no key-pair attribute to each other (either during Login or through the Text Command/Response mechanism).

```
   Byte /      0        |       1       |       2       |       3       |
       /                |               |               |               |
      |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
     +---------------+---------------+---------------+---------------+
    0| Opcode (0x46) | Reserved (0)                                  |
     +---------------+---------------+---------------+---------------+
    4| Length                                                        |
     +---------------+---------------+---------------+---------------+
    8| Reserved (0)                                                  |
```

```
  +                                                               +
12|                                                               |
  +---------------+---------------+---------------+---------------+
16| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
20| Desired Data Transfer Length                                  |
  +---------------+---------------+---------------+---------------+
24| Buffer Offset                                                 |
  +---------------+---------------+---------------+---------------+
28| ExpCmdRN                                                      |
  +---------------+---------------+---------------+---------------+
32| MaxCmdRN                                                      |
  +---------------+---------------+---------------+---------------+
36| Target Transfer Tag                                           |
  +---------------+---------------+---------------+---------------+
40/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
48
```

3.9.1. Desired Data Transfer Length and Buffer Offset

The target specifies how many bytes it wants the initiator to send as a result of this RTT message. The target may request the data from the initiator in several chunks, not necessarily in the

original order of the data. The target, therefore, also specifies a Buffer Offset indicating the point at which the data transfer should begin, relative to the beginning of the total data transfer.

3.9.2. Target Transfer Tag

The target assigns its own tag to each RTT request that it sends to the initiator. This can be used by the target to easily identify data it receives.

3.10. SCSI Data

The typical data transfer specifies the length of the data payload, the Transfer Tag provided by the receiver for this data transfer, and a buffer offset. The typical SCSI Data packet for WRITE (from initiator to target) has the following format:

```
Byte /    0          |        1          |        2          |        3          |
    /                |                   |                   |                   |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x05) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8| Buffer Offset                                                 |
   +---------------+---------------+---------------+---------------+
 12| Transfer Tag                                                  |
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Reserved (0)                                                  |
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 28| CmdRN                                                         |
   +---------------+---------------+---------------+---------------+
 28| ExpStatRN                                                     |
   +---------------+---------------+---------------+---------------+
 32/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48/ Payload                                                       /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

The typical SCSI Data packet for READ (from target to initiator) has the following format:

```
    Byte /    0          |        1        |        2        |        3        |
        /                |                 |                 |                 |
       |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
       +---------------+---------------+---------------+---------------+
      0| Opcode (0x45) | (0)     |S|O|U| Reserved (0)                  |
       +---------------+---------------+---------------+---------------+
      4| Length                                                        |
       +---------------+---------------+---------------+---------------+
      8| Buffer Offset                                                 |
       +---------------+---------------+---------------+---------------+
     12| Transfer Tag                                                  |
       +---------------+---------------+---------------+---------------+
     16| Initiator Task Tag                                            |
       +---------------+---------------+---------------+---------------+
     20| Residual Count                                                |
       +---------------+---------------+---------------+---------------+
     24| StatRN                                                        |
       +---------------+---------------+---------------+---------------+
     28| ExpCmdRN                                                      |
       +---------------+---------------+---------------+---------------+
     32| MaxCmdRN                                                      |
       +---------------+---------------+---------------+---------------+
     36| Command Status|iSCSI Status   | Reserved (0)                  |
       +---------------+---------------+---------------+---------------+
     40/ Reserved (0)                                                  /
      +/                                                               /
       +---------------+---------------+---------------+---------------+
     48/ Payload                                                       /
      +/                                                               /
       +---------------+---------------+---------------+---------------+
```

3.10.1. Length

The length field specifies the total number of bytes in the following payload.

3.10.2. Transfer Tag

The Transfer Tag identifies the target or initiator entity to which this data transfer relates. When the transfer is from the target to the initiator, the Transfer Tag is the Initiator Task Tag that was sent with the SCSI command. When the transfer is from the initiator to the target, the Transfer Tag is the Target Transfer Tag when RTT is enabled, or the Initiator Task Tag when RTT is disabled.

3.10.3. Buffer Offset

The Buffer Offset field contains the offset of the following data against the complete data transfer. The sum of the buffer offset and length should not exceed the expected transfer length for the command.

3.10.4. Flags

The last SCSI Data packet sent from a target to an initiator for a particular SCSI command that completed successfully may optionally also contain the Command Status for the data transfer. In this case Sense Data cannot be sent together with the Command Status. If the command completed with an error, then the response and sense data must be sent in a SCSI Response packet and must not be sent in a SCSI Data packet.

```
          Byte 1
     b0-1 as in an ordinary SCSI Response
     b2   (S) set to indicate that the Command Status field con-
          tains status.
     b3-7 not used (should be set to 0).
```

If the (S) bit is set, then there is meaning to the extra fields in the SCSI Data packet (StatRN, Command Status, Residual Count).

3.10.5. Packet numbering (CmdRN and StatRN)

On both inbound and outbound data the source may decide to number (sequence) the data packets to enable shorter recovery on connection failure. In case the source numbers data packets the destination is required to acknowledge them the same way it does with command and status packets - i.e. specifying the next expected packet.

3.11. Text Command

The Text Command is provided to allow the exchange of information and for future extensions. It permits the initiator to inform a target of its capabilities or to request some special operations.

```
Byte /    0          |       1         |       2         |       3         |
    /                |                 |                 |                 |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x04) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
```

```
   8/ Reserved (0)                                                    /
   +/                                                                 /
    +---------------+---------------+---------------+---------------+
  16| Initiator Task Tag                                            |
    +---------------+---------------+---------------+---------------+
  20| Reserved (0)                                                  |
    +---------------+---------------+---------------+---------------+
  24| CmdRN                                                         |
    +---------------+---------------+---------------+---------------+
  28| ExpStatRN                                                     |
    +---------------+---------------+---------------+---------------+
  32/ Reserved (0)                                                    /
   +/                                                                 /
    +---------------+---------------+---------------+---------------+
  48/ Text                                                            /
   +/                                                                 /
    +---------------+---------------+---------------+---------------+
```

3.11.1. Length

The length, in bytes, of the Text field.

3.11.2. Initiator Task Tag

The initiator assigned identifier for this Text Command.

3.11.3. Text

The initiator sends the target a set of key:value pairs in UTF-8 Unicode format. The key and value are separated by a ':' (0x3A) delimiter. Many key:value pairs can be included in the Text block by separating them with null ' ' (0x00) delimiters. Some basic key:value pairs are described in Appendix B. The target responds

by sending its response back to the initiator. The target and initiator can then perform some advanced operations based on their common capabilities.

Manufacturers may introduce new keys by prefixing them with their (reversed) domain name, for example,

    com.foo.bar.do_something:0000000000000003

Any key that the target does not understand may be ignored without affecting basic function. Once the target has processed all the key:value pairs, it responds with the Text Response command, listing the parameters that it supports. It is recommended that Text operations that will take a long time should be placed in their own Text command. If the Text Response does not contain a key that was requested, the initiator must assume that the key was not understood by the target.

3.12. Text Response

The Text Response message contains the responses of the target to the initiator's Text Command. The format of the Text field matches that of the Text Command.

```
Byte /      0        |        1        |        2        |        3        |
    /                |                 |                 |                 |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x44) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8/ Reserved (0)                                                  /
 +/                                                                /
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 24| StatRN                                                        |
   +---------------+---------------+---------------+---------------+
 28| ExpCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 32| MaxCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 36/ Reserved (0)                                                  /
 +/                                                                /
   +---------------+---------------+---------------+---------------+
 48/ Text Response                                                 /
 +/                                                                /
   +---------------+---------------+---------------+---------------+
```

3.12.1. Length

The length, in bytes, of the Text Response field.

3.12.2. Initiator Task Tag

The Initiator Task Tag matches the tag used in the initial Text Command and is used by the initiator to relate the Text Commands with the appropriate Text Responses.

3.12.3. Text Response

The Text Response field contains responses in the same key:value

format as the Text Command. Appendix B lists some basic Text Commands and their Responses. If the Text Response does not contain a key that was requested, the initiator must assume that the key was not understood by the target.

3.13. Login Command

After establishing a TCP connection between an initiator and a target, the initiator should issue a Login Command to gain further access to the target's resources.

```
Byte /      0          |        1          |        2          |        3          |
    /                  |                   |                   |                   |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x03) | Login Type    | Reserved (0)                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8| CID                           | RecoverCID or 0               |
   +---------------+---------------+---------------+---------------+
 12| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 16| ISID                          |TSID                           |
   +---------------+---------------+---------------+---------------+
 24| InitCmdRN   or   0                                            |
   +---------------+---------------+---------------+---------------+
 28/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48/ Login Parameters in Text Command Format                       /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

3.13.1. Login Type

Five types of logins are supported: clear text, RSA (Rivest Shamir Adelman) one way (to authenticate the client only), RSA two way (to authenticate both the server and the client), and implicit (in which a separate security protocol provides the credentials). The parameter "Access-ID" [AC] is used to identify the specific initiator.

0 no authentication
1 implicit
2 clear text password authentication
3 RSA 1 way
4 RSA 2 way

3.13.2. CID

A unique id for this connection within the session

3.13.3. RecoverCID

For a connection used to recover a lost TCP connection the initiator provides the CID of the failed connection. A simple target may reject recovery. In this case the initiator will terminate all outstanding commands with a check condition. Further action is up to

the initiator (implementation dependent).

3.13.4. Login Parameters

The initiator may provide some basic parameters in order to enable the target to determine if the initiator may in fact use the target's resources. The format of the parameters is as specified for the Text Command. Targets may require keys to indicate the Domain Name of the initiator and the target, and perhaps also an Authenticator key. The initiator may also provide additional parameters to the target in Text Command format, if the initiator so desires. Keys and their explanations are listed in Appendix B. Whenever desired an initiator will identify its view of the target as in:

    Target:<domain-name>[/modifier][:port]

implying that the target is known as:

    scsi://<domain-name>[/modifier]

and it should be connected through port "port" (the default, well known port, has an ICANN defined value of xx). Initiators can use the same type of naming implying machine and an optional principal (e.g. operating system image) as in:

    Initiator:<domain-name>[/modifier]

implying that the initiator is known as:

    iSCSI://<domain-name>[/modifier]

Thus the parameters passed for a clear-text password authentication are:

    Initiator:<domain-name>[/modifier]
    Target:<domain-name>[/modifier]
    Authenticator:open-sesame
    Access-ID:value

The modifier iSCSI-SYS is reserved for administrative functions.

ISID and TSID form collectively the SSID (session id). A TSID of 0 indicates a leading connection. Only a leading connection login can carry session specific parameters, e.g. max-connections-requested, the maximum immediate data length requested, etc..

3.13.5. InitCmdRN

Is significant only if TSID is 0 and indicates the starting Command reference number for this session; it should be 0 for all other instances.

3.13.6. Clear-text login

In clear text login, both the Access ID and the secret are not encrypted, and the target will either accept or reject the login.

3.13.7. RSA One Way Authentication

In RSA one way authorization, the initiator logs-on specifying the Access-ID. The target responds with a randomly generated string (not to exceed 64 bytes) that has been encrypted using the initiator's RSA public key, which is passed as the "Authentication-Initiator-Challenge" parameter. The initiator decrypts the string using its private key and returns the response as "Authentication-Initiator-Response" parameter. The target either accepts or rejects the login.

3.13.8. RSA Two Way authentication

In RSA two way authentication, the initiator generates a random string (not to exceed 64 bytes) that is encrypted with the RSA public key of the target. This string is sent to the target as the "Authenticator-Target-Challenge" parameter. The target decrypts this parameters and sends it back as the "Authentication-Target-Response", as well as generating an "Authentication-Initiator-Challenge" parameter as outlined above

3.13.9. Implicit authentication

In implicit authentication, both the Access ID and any authentication is performed by the security protocol, and no explicit Access ID or authentication information is exchanged during the login process.

3.13.10. TLS Session

The initiator can request a TLS session by entering a single text parameter of "Encrypted Session" set to the value of "TLS 1.0".

After the target has acknowledged the login command, a TLS 1.0 session is started on the same connection. Authentication is then performed using a second login command in clear text mode. The use of any specific set of certificates is not controlled by this specification.

3.14. Login Response

The target responds to the Login Command with a Login Response. It is sufficient for the target to respond with a Status indicating that the Login is accepted. In fact, the target may completely ignore the parameters that were sent to it and may provide service to any initiator that connects to it. The target may also return parameters using the format of the Text Response opcode, if it so desires. In particular, the target may want to provide its Authenticator key, so that the initiator can be sure that it is in fact talking with the correct target.

```
Byte /      0       |       1       |       2       |       3       |
    /               |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x43) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 16| ISID                          | TSID                          |
   +---------------+---------------+---------------+---------------+
 20| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 24| InitStatRN  or 0                                              |
   +---------------+---------------+---------------+---------------+
 28| ExpCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 32| MaxCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 36| Status        | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
 40/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48/ Login Parameters in Text Command Format                       /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

3.14.1. InitStatRN

Is significant only if TSID is 0 and indicates the starting status reference number for this session; it should be 0 for all other instances.

3.14.2. Status

The Status returned in a Login Response is one of the following:

0 accept login (will now accept SCSI commands)
1 reject login
2 additional authentication required
3 reject recovery

In the case that the Status is "accept login" the initiator may proceed to issue SCSI commands. In the case that the Status is "reject login" the initiator should immediately close down its end of the TCP connection, thus freeing up the target's port for some other connection. The target also has the option of immediately closing down its end of the TCP connection. In the case that the Status is "additional authentication required" the initiator must provide additional authentication information by issuing the Text Command with the appropriate key:value pairs; this may be required if the authentication method is based on a challenge/response algorithm. Upon receipt of the necessary authentication, the target will issue a Login Response with the "accept login" Status. SCSI Commands will not be accepted until the target provides a Login Response with the "accept login" Status. The TSID is an initiator identifying tag set by the target. A 0 in the returned TSID indicates that either the target supports only a single connection or that the ISID has already been used as a leading ISID. In both cases the target is rejecting the login.

3.15. Ping Command

```
Byte /      0       |       1       |       2       |       3       |
    /               |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x09) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 24| CmdRN                                                         |
   +---------------+---------------+---------------+---------------+
 28| ExpStatRN                                                     |
   +---------------+---------------+---------------+---------------+
 32/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48/ Ping Data (optional)                                          /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

The Ping Command can be used to verify that a connection is still active and all it's components are operational; unlike the NOP message Ping has a Task Tag and can be delivered in order. It may be useful in the case where an initiator has been waiting a long time for the response to some command, and the initiator suspects that there is some problem with the connection. When a target receives the Ping Command, it should respond with a Ping Response, duplicating as much of the data as possible that was provided in the Ping Command (if such data was present). If the initiator does not receive the Ping Response within some period of time (determined by the initiator), or if the data returned by the Ping Response is different from the data that was in the Ping Command, the initiator may conclude that there is a problem with the connection. The initiator will then close the connection and may try to establish a new connection.

3.15.1. Length

The length of the optional Ping Data.

3.15.2. Initiator Task Tag

An initiator assigned identifier for the operation.

3.15.3. Ping Data

Binary data that will be reflected in the Ping Response.

3.16. Ping Response

```
Byte /    0           |       1       |       2       |       3       |
    /                 |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x49) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8/ Reserved (0)                                                  /
 +/                                                                /
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 24| StatRN                                                        |
   +---------------+---------------+---------------+---------------+
 28| ExpCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
```

```
   32| MaxCmdRN                                                      |
     +---------------+---------------+---------------+---------------+
   36/ Reserved (0)                                                  /
    +/                                                               /
     +---------------+---------------+---------------+---------------+
   48/ Return Ping Data                                              /
    +/                                                               /
     +---------------+---------------+---------------+---------------+
```

When a target receives the Ping Command, it should respond with a Ping Response, duplicating the data and Initiator Task Tag that was provided in the Ping Command, if present.

3.17. Map Command

```
   Byte /    0          |       1       |       2       |       3       |
       /                |               |               |               |
      |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
      +---------------+---------------+---------------+---------------+
     0| Opcode (0x0a) | Function      | Reserved (0)                  |
      +---------------+---------------+---------------+---------------+
     4| Length                                                        |
      +---------------+---------------+---------------+---------------+
     8| Reserved (0)                                                  |
      +                                                               +
    12|                                                               |
      +---------------+---------------+---------------+---------------+
    16| Initiator Task Tag                                            |
      +---------------+---------------+---------------+---------------+
    20| Reserved (0)                                                  |
      +---------------+---------------+---------------+---------------+
    24| CmdRN                                                         |
      +---------------+---------------+---------------+---------------+
    28| ExpStatRN                                                     |
      +---------------+---------------+---------------+---------------+
    32/ Reserved (0)                                                  /
     +/                                                               /
      +---------------+---------------+---------------+---------------+
    48| Descriptor Type               | Descriptor Length             |
      +---------------+---------------+---------------+---------------+
    52/ Descriptor                                                    /
     +/                                                               /
      +---------------+---------------+---------------+---------------+
   --------------------------------------------------------------------
      +---------------+---------------+---------------+---------------+
      | Descriptor Type               | Descriptor Length             |
```

```
   +---------------+---------------+---------------+---------------+
   / Descriptor                                                    /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

or


```
   +---------------+---------------+---------------+---------------+
 48| 8 byte Descriptor                                             |
  +|                                                               |
   +---------------+---------------+---------------+---------------+
------------------------------------------------------------------------
   +---------------+---------------+---------------+---------------+
 N | 8 byte Descriptor                                             |
  +|                                                               |
   +---------------+---------------+---------------+---------------+
```

The mapping command enables the initiator to map iSCSI specific addresses and access control information into formats compliant with the SCSI command standards (e.g., [SPC-2]).

3.17.1. Function

Two functions are required for mapping:

1 Map - given an address or access control information provide the 8 byte SCSI compliant address reference

0 Unmap Given a SCSI compliant address reference remove the mapping associated with it.

Address/access control descriptors follow the header. For the map function the following descriptor types are defined:

0 IP Version 4 TCP address followed by a TAN(Target Acquired Name); length should be 6 + the view length + 1

1 IP Version 6 TCP address followed by a TAN; length should be 18 bytes + the view length + 1

2 iSCSI URL (domain name terminated with null followed by a TAN followed by null)

3 FC address & port - in case access control is based on transport ID

4 access proxy token

Details for 3 & 4 have to be coordinated with T10

For the unmap function the descriptors are standard 8 byte SRAs (SCSI Reference Address)

3.18. Map Response

```
Byte /      0       |       1       |       2       |       3       |
    /               |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x4a) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8| Reserved (0)                                                  |
   +                                                               +
 12|                                                               |
   +---------------+---------------+---------------+---------------+
 16| Initiator Task Tag                                            |
   +---------------+---------------+---------------+---------------+
 20| Reserved (0)                                                  |
   +---------------+---------------+---------------+---------------+
 24| StatRN                                                        |
   +---------------+---------------+---------------+---------------+
 28| ExpCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 32| MaxCmdRN                                                      |
   +---------------+---------------+---------------+---------------+
 36| Response      | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
 40/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
 48
```

The target provides a Response which may take on the following values:

    0   Function Complete
    1   Map Function Rejected - Bad Descriptors
    2   Map Function Rejected - too many descriptors
    3   Unmap Function Rejected - Bad Descriptor

If the Response to a map is function complete the data following the header contains the SRAs to be used in third party commands; each SRA matches a descriptor in the Map command.

3.19. Third Party Commands

There are some third-party SCSI commands, such as (EXTENDED)COPY and COMPARE, that involve more than one target. In its most general form those commands involve the "original target" called the COPY-Manager and a (variable) number of other machines called source and destination. The whole operation is described by one "master CDB" delivered to the Copy manager and a series of descriptor blocks; each descriptor block addresses a source and destination target and LU and a description of the work to be done in terms of blocks or bytes as required by the device types. The relevant SCSI standards do not require full support of the (EXTENDED) COPY or COMPARE nor do they provide a detailed execution model. We will assume, in the spirit of [SPC-2] that a COPY manager will read data from a source and write them to a destination.

To address them an iSCSI COPY manager will use information provided to it through map commands and the SRAs and flags provided in the descriptors - allowing for iSCSI and FC sources and destinations.

Enabling a FC COPY manager to support iSCSI sources and destinations is subject to coordination with T10.

3.20. Opcode Not Understood

```
Byte /      0       |       1       |       2       |       3       |
    /               |               |               |               |
   |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
   +---------------+---------------+---------------+---------------+
  0| Opcode (0x48) | Reserved (0)                                  |
   +---------------+---------------+---------------+---------------+
  4| Length                                                        |
   +---------------+---------------+---------------+---------------+
  8/ Reserved (0)                                                  /
  +/                                                               /
   +---------------+---------------+---------------+---------------+
```

```
48/ Header of Bad Message                                        /
 +/                                                              /
  +---------------+---------------+---------------+---------------+
96
```

It may happen that a target receives a message with an Opcode that it doesn't recognize. This may occur because of a new version of the protocol that defines a new Opcode, or because of some corruption of a message header. The target returns the header of the message with the unrecognized opcode as the data of the response.

4. iSCSI Error Handling

4.1. Communications Errors

For any outstanding SCSI command, it is assumed that iSCSI in conjunction with SCSI at the initiator is able to keep enough information to be able to rebuild the command PDU, that outgoing data is available (in host memory) for retransmission while the command is outstanding. It is also assumed that at a target iSCSI and specialized TCP implementations are able to recover unacknowledged data packets from a closing connection or, alternatively the target has means to re-read data from a device-server. It is further assumed that a target will keep the "status & sense" for commands it has executed while the total number of outstanding commands and executed commands does not exceed its limit. A target will sequentially number the delivered responses and thus enable initiators to tell when a response is missing and which response is missing.

Under those conditions, iSCSI will be able to keep a session in operation provided that it is able to keep/establish at least one TCP connection between the initiator and target in a timely fashion. Unfortunately the maximum admissible recovery time is a function of the target and for some devices and communications networks recovery may be complex and may percolate to upper software layers. It is assumed that targets and/or initiators will recognize a failing connection by either transport level means (TCP) or

by a gap in the command or response stream that does not get filled for a long time, or by a failing iSCSI ping (the later should be used periodically by highly reliable implementations). The recovery involves the following steps:

- -abort offending TCP connection(s) (target & initiator) and recover at target all unacknowledged read-data
- -create one or more new TCP connections (within the same session) and associate all outstanding commands from the failed connection to the new connection at both initiator and target.
- -the initiator will reissue all outstanding commands with their original Initiator Task Tag and their original CmdRN if they are not acknowledged yet or a new CmdRN if they where acknowledged; in any case the retry (X) flag in the command PDU will be set
- -upon receiving the new/retry commands the target will resume command execution; for write commands it means requesting data retransmission through RTT, for reads retransmitting recovered data and for "terminated" commands retransmitting the status & sense while retaining the original StatRN. If data recovery is not possible the target will either provide data from the media or redo the operation (if the operation is not idempotent the device server may fail the operation).

4.2. Protocol Errors

The authors recognize that mapping framed messages over a "stream" connection (like TCP) makes the proposed mechanisms vulnerable to simple software framing errors and introducing framing mechanisms may be onerous for performance and bandwidth. Command reference numbers and the above mechanisms for connection drop and reestablishment will help handle this type of mapping errors.

4.3. Session Errors

If all the connections of a session fail and can't be reestablished in a short time or if initiators detect protocol errors repeatedly an initiator may choose to terminate a session and establish a new session (indicating old session termination?). It will terminate all outstanding requests with an iSCSI error indication before initiating a new session. A target that detects one of the above errors will take the following actions:

- Reset the TCP connections (close the session).
- Abort all Tasks in the task set for the corresponding initiator.

5. Notes to Implementers

This section notes some of the performance and reliability considerations of the iSCSI protocol. This protocol was designed to efficient silicon and software implementations. The iSCSI tag mechanism was designed to enable RDMA at the iSCSI level or lower.

5.1. Small TCP Segments

It is recommended that TCP segments be limited in size to no more than 8K bytes. One reason we recommend small segments is to allow a stronger type of checksum, possibly utilizing CRC, which is practical only for smaller segments.

5.2. Multiple Network Adapters

The iSCSI protocol allows multiple connections, not all of which need go over the same network adapter. If multiple network connections are to be utilized with hardware support, the iSCSI protocol command-data-status allegiance to one TCP connection insure that there is no need to replicate information across network adapters or otherwise require them to cooperate.

5.3. Autosense

Autosense refers to the automatic return of sense data to the initiator in case a command did not complete successfully. If autosense is turned off, the initiator must explicitly request that sense data be sent to it after some command has completed with a CHECK CONDITION status. The default for iSCSI is to work with Autosense enabled. Note that even if a SCSI target/LUN does not support Autosense, it may still be possible for iSCSI to work with Autosense. This can be accomplished as follows. Whenever a CHECK CONDITION status is about to be returned, the iSCSI component on the target immediately queries the target/LUN for the sense data. iSCSI can then return the sense data to the initiator together with the CHECK CONDITION status. It is not necessary for iSCSI to wait for the initiator to explicitly request the sense data; the target iSCSI code can perform this operation automatically, even for devices/LUNs that do not ordinarily provide automatic sense data.

5.4. TCP Connection Options

Some targets may want to inform (or negotiate with) an initiator concerning some parameters related to bandwidth, Quality of Service, or some other available features on its various network connections. These are exchanged between the initiator and the target using Text Commands and Responses.

6. Security Considerations

6.1. Data Integrity

We assume that end-to-end data integrity can be assured by TCP, by adding a more powerful checksum option whenever this is considered important, or leaving the standard checksum for those applications in which data integrity is not of utmost importance.

6.2. Network operations and the Threat Model

Historically, native storage systems have not had to consider security due to the fact that their environments offered minimal security risks. That is, these environments consisted of storage devices either directly attached to hosts or connected via a subnet distinctly separate from the communications network. The use of storage protocols, such as SCSI, over IP networks requires that security concerns be addressed.

6.2.1. Threat Model

Attacks fall into three main areas; passive, active, and denial of service.

6.2.1.1. Passive Attacks

In general, data transfers will be made through a switched fabric, making sniffing difficult. Also, the nature of the data (block transfers), even if sniffed, would not necessarily be readily understandable to the attacker. That being said, a determined attacker could, by capturing of content and analyzing traffic over time, could replicate enough of a drive to make the captured data meaningful. Certain storage operations which are mostly unidirectional, such as writing to a tape or reading from a CD-ROM, are even more susceptible to passive attacks since the listener will be able to replicate most if not all of the operation.

Passive attacks by traffic analysis alone is deemed out of scope since it is unlikely that the listener will be able to guess any pertinent information without knowing the content of the messages. It is also out of scope to detect passive attacks. The protocol must be able to prevent passive attacks by masking the contents of messages through some form of encryption.

Finally, it is assumed that a strong authentication mechanism will be necessary. Therefore, any long-lived passwords or private keys must never be sent in the clear.

6.2.1.2. Active Attacks

Whereas passive attacks involve SNIFFING, active attacks will generally involve SPOOFING. If an attacker can successfully masquerade as a client, he will have total read/write access to those storage resources assigned to that client. Spoofing as a server is more

difficult, since many operations involve client reads of some expected or otherwise understandable data.

Most likely, many of the sessions will be long-lived. This feature has a dual effect of making these sessions more vulnerable to attack (hijacking TCP connections, cryptographic attacks), while at the same time providing mechanisms to detect attacks. An attempt to open a session while one is already active can be treated as a possible attack. Both the transport and session layer protocols will have sequencing that would need to be adhered to by the attacker to avoid generating errors that could also be treated as a possible attack.

Message modification can be a significant threat to an environment reliant on the integrity of the data. Message replay, insertion, or deletion will generally produce errors (such as data overruns/underruns) that can be recovered successfully, they can have the effect of reducing performance, and as such can act as a denial of service. It is possible that an attacker can modify a message in such a way the session becomes out of sync, resulting in a tear down of the session.

6.2.2. Security Model

6.2.2.1. No Security

This mode does not authenticate nor does it encrypt data. This mode should only be used in environments where there is minimal security risk and little chance for configuration errors.

6.2.2.2. End-to-End Authentication

This mode protects against an unauthorized access to storage resources either through an active attack (SPOOFING) or configuration errors. Once the client is authenticated, all messages are sent and received in the clear. This mode should only be used when there is minimal risk to man-in-the-middle attacks, eavesdropping, message insertion, deletion, and modification. For example, this mode can be used when IPsec is used in security gateways.

6.2.2.3. Encryption

This mode provides for the end-to-end encryption (e.g. IPsec). In addition to authenticating the client, it provides end-to-end data integrity and protects against man-in-the-middle attacks, eavesdropping, message insertion, deletion, and modification.

A connection or multiple connections can be protected end-to-end by using IPSec . In this case, the initiator must use the "Implicit Authentication" parameter to indicate that IPSec should be used to specify the Access ID and perform authentication.

6.2.3. Other Considerations

Due to long-lived sessions, is there a need for periodic authentication after the session is established? For example, should the client be challenged during key-alive exchanges in addition to login?

Due to long-lived sessions with encryption, is there a higher level of vulnerability to cryptographic attacks?

6.3. Login Process

In some environments, a target will not be interested in authenticating the initiator. In this case, the target can simply ignore some or all of the parameters sent in a Login Command, and the target can simply reply with a basic Login Response indicating a successful login. Some targets may want to perform some kind of authentication. The Authenticator key is defined for this purpose. Various authentication schemes can be used, including encrypted passwords and trusted certificate authorities. Once the initiator and target are confident of the identity of the attached party, the established channel is considered secure. It is anticipated that most target devices will not bother with all of the possible checks, but the protocol provides sufficient means to perform the checks, if required by the target.

6.4. ICANN Considerations

There will be a well known port for iSCSI connections. This well known port will have to be registered with ICANN.

A checksum type will also have to be registered with ICANN.

7. Authors' Addresses

Julian Satran
Kalman Meth
IBM, Haifa Research Lab
MATAM - Advanced Technology Center
Haifa 31905, Israel
Phone +972 4 829 6211
Email: Julian_Satran@vnet.ibm.com meth@il.ibm.com

Daniel F. Smith
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099, USA
Phone: +1 408 927 2072
Email: dfsmith@almaden.ibm.com

Costa Sapuntzakis
Cisco Systems, Inc.
170 W. Tasman Drive
San Jose, CA 95134, USA
Phone: +1 408 525 5497
Email: csapuntz@cisco.com

Randy Haagens
Hewlett-Packard Company
8000 Foothills Blvd.
Roseville, CA 95747-5668, USA
Phone: +1 (916) 785-4578
E-mail: Randy_Haagens@hp.com

Efri Zeidner
SANGate
Israel
efri@sangate.com

Paul von Stamwitz
Adaptec, Inc.
691 South Milpitas Boulevard
Milpitas, CA 95035
Phone:(408) 957-5660
E-mail: paulv@corp.adaptec.com

Luciano Dalle Ore
Quantum Corp.
Phone: +1 (408) 232 6524
E-mail: luciano.dalle.ore@quantum.com

Comments may be sent to Julian Satran

8. References and Bibliography

[SAM2] ANSI X3.270-1998, SCSI-3 Architecture Model (SAM-2)
[TLS] The TLS Protocol, RFC 2246, T. Dierks et al.

[ALTC] Internet Draft: Alternative checksums (work in progress)
[CAM] ANSI X3.232-199X, Common Access Method-3 (Cam-3)
[CRC] ISO 3309, High-Level Data Link Control (CRC 32)
[RFC793] Transmission Control Protocol, RFC 793
[RFC1122] Requirements for Internet Hosts-Communication Layer, RFC 1122, R. Braden (editor)
[SBC] ANSI X3.306-199X, SCSI-3 Block Commands (SBC)
[SCSI2] ANSI X3.131-1994, SCSI-2
[SPC] ANSI X3.301-199X, SCSI-3 Primary Commands (SPC)
[AC] A detailed proposal for Access Control, Jim Hafner, T10/99-245

9. Appendix A - Examples

9.1. Read operation example

Initiator Function	Message Type	Target Function
Command request (read)	SCSI Command (READ)>>>	
		Prepare Data Transfer
Receive Data	<<< SCSI Data	Send Data
Receive Data	<<< SCSI Data	Send Data
Receive Data	<<< SCSI Data	Send Data
	<<< SCSI Response	Send Status and Sense
Command Complete		

9.2. Write operation example

Initiator Function	Message Type	Target Function
Command request (write)	SCSI Command (WRITE)>>>	Receive command and queue it
		Process old commands

	<<< RTT	Ready to process WRITE command
Send Data	SCSI Data >>>	Receive Data
Send Data	SCSI Data >>>	Receive Data
	<<< RTT	
Send Data	SCSI Data >>>	Receive Data
	<<< SCSI Response	Send Status and Sense
Command Complete		

10. Appendix B - Login/Text keys

10.1. Target

Target:domainname[/modifier]

Examples:

```
Target:disk-array.sj-bldg-h.cisco.com
Target:disk-array.sj-bldg-h.cisco.com/disk3
```

This key is provided by the initiator of the TCP connection to the remote endpoint. The Target key specifies the domain name of the target, since that information is not available from the TCP layer. The target is not required to support this key. The initiator should send this key in the first login message. The Target key might be used by the target to learn the intended initiator view of the target.

10.2. Initiator

Initiator:[domainname[/modifier]] Examples:

Initiator:sample.foobar.org
Initiator:cluster.foobar.org/machine1
Initiator:

The Initiator key enables the initiator to identify itself to the remote endpoint. The domain name should be that of the initiator. A zero-length domain name is interpreted as "other side of TCP connection". The target may silently ignore this key if it does not support it. For more security, a certificate-based protocol [TLS] may be used on the channel and take precedence over this protocol.

10.3. Authenticator

Authenticator:<UTF8-String> Examples:

Authenticator:open-sesame

The authenticator is a secret that the initiator uses to gain access to the target's LUNs.

10.4. SendAuthenticator

SendAuthenticator:yes Response: Authenticator:<UTF8-String> Examples:

SendAuthenticator:yes
-> Authenticator:alakazam

The SendAuthenticator key is used to request from the party on the other side of the TCP connection to send its Authenticator. iSCSI devices may refuse to grant access until proper authentication has been performed by the parties involved.

10.5. UseRTT

UseRTT:<yes|no> Response: UseRTT:<yes|no> Examples:

UseRTT:no
-> UseRTT:no

The UseRTT key is used to turn off the default use of RTT, thus allowing an initiator to send data to a target without the target having sent an RTT to the initiator. The default action is that RTT is required, unless both the initiator and the target send this key-pair attribute specifying UseRTT:no. Once UseRTT has been set to 'no', it cannot be set back to 'yes'.

Access-ID:<accessid-in-text-format>

10.6. Authentication-Initiator-Challenge:<aleph>

10.7. Authentication-Initiator-Response:<beth>

10.8. Authentication-Target-Challenge:<gimel>

10.9. Authentication-Target-Response:<dalet>